Ridesharing on Timetabled Transport Services: A Multiagent Planning Approach

Unspecified Journal
Volume 00, Number 0, Pages 000–000
S ????-????(XX)0000-0

RIDESHARING ON TIMETABLED TRANSPORT SERVICES: A MULTIAGENT PLANNING APPROACH

JAN HRNČÍŘ, MICHAEL ROVATSOS, AND MICHAL JAKOB

Abstract. Ridesharing, i.e., the problem of finding parts of routes that can be shared by several travellers with different points of departure and destinations, is a complex, multiagent decision-making problem. The problem has been widely studied but only for the case of ridesharing using freely moving vehicles not bound to fixed routes and/or schedules – ridesharing on timetabled public transport services has not been previously considered. In this paper, we address this problem and propose a solution employing strategic multiagent planning that guarantees that for any shared journey plan found, each individual is better off taking the shared ride rather than travelling alone, thus providing a clear incentive to participate in it. We evaluate the proposed solution on real-world scenarios in terms of the algorithm's scalability and the ability to address the inherent trade-off between cost savings and the prolongation of journey duration. The results show that under a wide range of circumstances our algorithm finds attractive shared journey plans. In addition to serving as a basis for traveller-oriented ridesharing service, our system allows stakeholders to determine appropriate pricing policies to incentivise group travel and to predict the effects of potential service changes.

1. Introduction

Travelling is an important and frequent activity, yet people willing to travel have to face problems with rising fuel prices, carbon footprint and traffic jams. One way to tackle these problems is through *ridesharing*, i.e., purposeful and explicit planning to create groups of people travel together in a single vehicle for parts of the journey. Participants in such schemes can benefit from ridesharing in several ways: sharing parts of a journey may reduce cost (e.g., through group tickets), carbon footprint (e.g., when sharing a private car), and travellers can enjoy the company of others on a long journey.

In general, ridesharing is a widely studied problem – existing work, however, focuses exclusively on ridesharing using vehicles that can move *freely* on a road transport network. This overlooks the potential for innovative future transport schemes that might exploit ridesharing using *timetabled public transport*. Here, customised group discount schemes could be devised to balance the load across different times of the day, or to make more efficient use of the capacity of public means of transport. Also, joint travel can be used to increase the comfort and safety of individuals, e.g., for female travellers using night buses, or groups of schoolchildren. In more advanced scenarios, one could even imagine ridesharing on public means of transport being combined with working together while travelling,

2010 *Mathematics Subject Classification.* Primary.

holding meetings on the road, meeting people with common interests, etc. We would argue that, in fact, *any* future intelligent transport scheme for citizens that attempts to address the *social dimension* of travel will be incomplete if it does not take into account timetabled public transport.

Except for our own earlier work [27], no existing work seems to attempt to compute *joint* travel plans based on public transport timetables and geographical stop locations, let alone in a way that takes into account the *strategic* nature of the problem, which comes about through the different (and potentially conflicting) preferences of individual travellers. From the point of view of (multiagent) planning [13], i.e., the problem of synthesising sequences of actions to reach a certain goal – in this case, arrival at a destination from a given point of departure – for several travellers in parallel, ridesharing on timetabled services presents itself as a very complex application scenario: To begin with, even if one restricted oneself to centralised planning, the domain is huge – public transport data for the UK alone currently involves 240,590 timetable connections for trains and coaches (even excluding local city buses), which would have to be translated to a quarter of a million planning actions, at least in a naive formalisation of the domain. This is the case even if we assume a *non-strategic* setting, where individuals' preferences are not taken into account, and we are simply looking for a set of itinerary that gets everybody to their destination, without any regard for how costly this might be for the individual, or how the joint plan might favour some agents while putting others at a disadvantage. Moreover, considering a *strategic* setting where we are looking for a plan for multiple self-interested agents that are willing to cooperate only if it is beneficial for them is known to be exponentially harder than planning for each agent individually [5]. Yet any automated service that proposes joint journeys would have to guarantee such strategic properties in order to be acceptable for human users (who could then even leave it to the service to negotiate trips on their behalf).

In our previous paper [27], we discussed the possibility of using a pre-processing step to group users together in such a way that would permit applying our algorithm to thousands or even millions of users in a larger geographical area (e.g., an entire country). In this paper, we present an improved version of our algorithm which includes a pre-processing step that clusters likely co-travellers together based on the overall direction of their individual trips and the distance between the origin and destination points of these individual trips. We show that this pre-processing step enables us to reduce plan computation times from over an hour to a few minutes in the worst case, while still resulting in substantial benefits from ridesharing for those participating in shared journeys. The core of our algorithm is based on a domain-independent *best-response planning* [29] approach which is the only available planner that can solve strategic multiagent planning problems of the scale required, and whose properties and assumptions combine particularly well with the ridesharing problem in hand.

The contribution of our work is threefold: Firstly, we show that current multiagent planning technology can be used in important planning domains such as ridesharing by presenting its application to a practical problem that cannot be solved with other existing techniques. In the process, we describe the engineering steps that are necessary to deal with the challenges of real-world large-scale data and propose suitable solutions. Secondly, we present an algorithm that combines

different techniques in a practically-oriented way and works with real-world public transport timetables and realistic travel demand even though it is largely based on extensible, domain-independent, off-the-shelf heuristic problem solvers. Thirdly, we evaluate the proposed algorithm on such real-world data, taking public transport services of the Yorkshire region of the UK as an example. The evaluation not only analyses the performance of the proposed approach but also provides insights into general relationships between journey duration and cost in realistic ridesharing scenarios.

We start off with an overview of the related work in section 2. This is followed by a formal specification of the timetabled transport ridesharing problem in section 3 based on the model used in [29]. Section 4 introduces our four-phase algorithm for strategic planning in ridesharing domains. An extensive experimental evaluation of the algorithm is presented in section 5. Section 6 presents a discussion of our results and section 7 concludes.

2. Related Work

Ridesharing is a long known and widely studied problem – existing work, however, focuses exclusively on ridesharing using vehicles that can move freely on a road transport network, without schedule or route restrictions. The work on such *non-timetabled* ridesharing covers the whole spectrum from formal problem models, through solution algorithms up to practical consumer-oriented services and applications.

On the theoretical side, the vehicle-based ridesharing problem is typically formalised as a *Dial-a-Ride Problem (DARP)*. Different variants of DARPs exist, differing, for example, in the nature of traveller's constraints, the distribution of pickup and delivery locations, the criteria optimised, or the level of dynamism supported. A comprehensive review of different variants of DARPs, along with a list of algorithmic solution approaches, is given by Cordeau et al. [10]. Most of the existing approaches rely on a centralised coordination entity responsible for collecting requests and producing vehicle assignment and schedules, though more decentralised approaches have also been presented more recently [43]. Bergbelia et al. [1] summarise recent advances in *real-time ridesharing*, which has been gaining prominence with the growing penetration of internet-connected smartphones and GPS-enabled vehicle localisation technologies. Existing work almost exclusively considers a single mode of transport only. One of few exceptions is the work of Horn et al. [25] which considers demand-responsive ridesharing in the context of flexible, multi-modal transport systems; the actual ridesharing is, however, only supported for demand-responsive non-timetabled journey legs. On the practical side, there exist various online services for car (e.g., liftshare.com or citycarclub.co.uk), bike, and walk sharing as well as services which assist users in negotiating shared journeys (e.g., companions2travel.co.uk, travbuddy.com).

Journey planning for timetabled public transport services has been extensively studied in the single-agent case. The problem is typically formalised as the *earliest arrival problem* with two major ways to represent public transport timetables for the planning algorithm as a search graph. A *time-expanded approach* [36] where each event at a stop, e.g., the departure of a train, is modelled as a node in the graph; and a *time-dependent approach* [7] where the graph contains only one node for each station. To speed up the search process, many speed-up techniques for

a basic shortest-path algorithm, e.g., Dijkstra's algorithm, have been proposed, including the *multi-level graph* approach [38], *access-node routing* [14], and *core-ALT* [35]. These algorithms are the basis of public travel planning services (e.g., in the UK, nationalrail.co.uk for trains, traveline.info and maps.google.com for multi-modal transport) that automate *individual* travel planning for one or several means of transport.

So although both ridesharing using freely moving vehicle and single-agent journey planning for timetabled services have been extensively studied, the combination of both, i.e., ridesharing on timetabled services, has not been – to the best of our knowledge – studied before (with the exception of our previous paper).

Automated planning technology [24] has developed a variety of scalable heuristic algorithms for tackling hard planning problems, where plans, i.e., sequences of actions that achieve a given goal from a given initial state, are calculated by domain-independent problem solvers. Unlike other approaches to route planning and ridesharing, automated planning techniques permit a fairly straightforward formalisation of travel domains, and allow us to capture the joint action space and complex cost landscape resulting from travellers' concurrent activities. In terms of algorithmic complexity, the kind of multiagent planning needed to compute ridesharing plans for several agents is significantly harder than single-agent planning for two reasons: Firstly, the ability of each agent to execute actions concurrently [2] may result result in exponentially large sets of actions available in each step in the worst case. Secondly, whenever individual agents have different (and potentially conflicting) goals [4], a joint solution must satisfy additional requirements, e.g., being compatible with everyone's individual preferences, or not providing any incentive for any individual to deviate from the joint plan. Solving the *general* multiagent planning for problem sizes of the scale we are interested in in real-world ridesharing is therefore not currently possible using existing techniques.

Because of the desire to integrate different travellers' individual plans, ridesharing is quite similar to plan merging (e.g. [23, 41], where individual agents' plans are incrementally integrated into a joint solution. Compared to these approaches, however, in our domain every agent can always achieve their plan regardless of what others do, and agents do not require others' "help" to achieve their goals. This makes the problem simpler than those of plan merging though, in return, we place *much* higher scalability demands on the problem than those authors, who typically evaluate their algorithms only on toy domains.

This explains also why, as will be shown below, we are able to achieve much higher scalability than state-of-the-art multiagent plan synthesis algorithms, e.g., [33, 17, 34]. These algorithms exploit "locality" in different ways in order to be able to plan for parts of a multiagent planning problem while temporarily ignoring others, e.g., by considering non-interacting subplans in isolation from each other.In a sense, our problem involves even more loosely coupled sub-tasks, as these can be essentially solved in a completely independent way, except in terms of cost optimisation.

The relationship between our work and approaches that focus more on decentralised planning, plan co-ordination, and conflict resolution among independent planning agents (e.g., [11, 12]) is similar – as no hard conflicts can arise among individual plans in ridesharing, it is not essential to co-ordinate individual plans with each other, other than for cost optimisation purposes.

As far as the strategic aspect is concerned, this is obviously also relevant to ridesharing as ultimately each co-traveller wants to achieve an optimal solution for themselves. Various approaches have studied this problem in the past (e.g. [18, 30, 4], yet none of them has been shown to scale to the type of domain we are interested in, with the exception of [29], which makes certain simplifying assumptions to achieve scalability: it does not consider *joint* deviation from equilibrium solutions (i.e. it only safeguards against individual agents opting out of a joint plan, not whole sub-groups of agents), and it assumes that agents will honour their promises when they have agreed on a joint plan. We believe that both these assumptions are reasonable in ridesharing, as we are envisioning a platform on which users would be automatically grouped together whenever a rideshare would be beneficial to each one of them. On such a platform, it is reasonable to assume that agreements could be enforced through a trusted third party, and that collusion among travellers could be avoided by not disclosing their identities to each other until the purchase of all tickets has been completed. Below, we describe how this algorithm serves as the basic planning method used in our ridesharing system.

3. Problem Formulation

Informally, the problem we are trying to solve is the following: Assume a (potentially very large) set of agents who represent individual travellers, with their individual trips specified in terms of source and target location. Assume also that agents want to optimise the individual utility accrued from a trip, and this utility may depend on travel cost and number of people travelling along each leg of the journey (generally, we will assume that group travel has a positive effect on utility, as we want to study the impact of this very aspect on travel behaviour). Based on this information, we are looking for an algorithm that can identify appropriate groups of travellers who could share parts of their journeys using the full timetabling information of public transport systems, and determine their precise joint travel plan. Also, we want to be sure that if we propose this plan to a group, none of the individual agents will have an incentive to improve on the proposed solution by deviating from it, i.e., we only want to suggest rideshares from which *all* travellers involved will benefit.

This section provides the formalisation of the *timetabled transport ridesharing problem*, which is then used by the ridesharing planning algorithm described in the next section. This formalisation builds on a representation of timetabled transport services captured at two different levels of granularity, which we call the *relaxed* and *full* transport services domain. From a planning perspective, problem formulation-builds on the definition of a multiagent planning problem, which is essentially the combination of several individual planning problems involving an initial and goal state, as well as a set of actions that can be performed by the agent, i.e. the public transport services she can use.

3.1. Timetabled Transport Services Representation.

Since the full travel planning domain with a full granularity of timetabled connections is too large for any current state-of-the-art planner to deal with, we distinguish the *full transport services domain* from what we call the *relaxed transport services domain*, which we will use to come up with an initial plan before mapping it to the full timetable information in our algorithm below. Roughly speaking, the relaxed domain contains information about all travel connections in the transport network with their

respective shortest travel times, and ignores any concrete service timetables and information about which passengers are using which sevices, which are only included in the full domain (the relaxed domain also ignores direct connections among locations with intermediate stops, for reasons that will be explained below). Since only trips that are possible in the relaxed domain are possible in the full domain, this gives us a sound relaxation of the problem we can work with. This relaxation is of course incomplete in the general case, as many trips that are possible in theory cannot be performed in practice due to timetabling constraints, both regarding transport services and participating travellers' requirements.

The *relaxed domain* is a single-agent planning domain represented as a weighted directed graph $T = (V, E, w)$ where the set of nodes V represents the stops and the set of edges E represents the connections provided by a service. The graph must be directed because there exist stops that can only be used in one direction . There is an edge $e = (A, B) \in E$ from stop A to B in this graph if there is at least one connection from A to B in the timetable. The weight $w(e)$ of this edge is given by the weight function $w : E \to \mathbb{R}_0^+$ which returns the minimal time needed for travelling from A to B. A plan $P_i = \langle A_1 \to A_2, A_2 \to A_3, \ldots, A_{k-1} \to A_k \rangle$ found in the relaxed domain for the agent i is a sequence of $k - 1$ connections to travel from its origin A_1 to its destination A_k.

A small example of the relaxed domain is shown in Figure 1. An example plan for an agent travelling from C to F is $P_1 = \langle C \to D, D \to E, E \to F \rangle$. To give an idea of the difference between the relaxed domain and the full timetable in terms of domain complexity, there are 497 connections in the relaxed domain for trains and coaches in the Yorkshire area compared to 10,295 timetabled, actual connections.

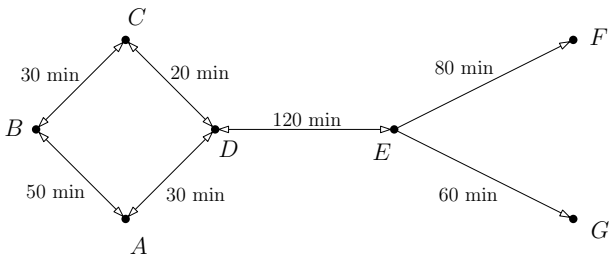


FIGURE 1. An example of the relaxed domain showing basic connection times (e.g., it takes 50 minutes to travel from A to B)

Direct trains that do not stop at every stop are filtered out from the relaxed domain for the following reason: Assume that in Figure 1, there is only one agent travelling from C to F and that its plan in the relaxed domain is to use a direct train from C to F. In this case, it is only possible to match its plan to direct train connections from C to F, and not to trains that stop at C, D, E, and F. Therefore, the agent's plan cannot be matched against all possible trains between C and F which is problematic especially in the case where the majority of trains stop at every stop and only a few trains are direct. On the other hand, it is possible to match a plan with a train stopping in every stop to a direct train, as explained later in section 4.4.

Assume a set $S = \{1, \ldots, n\}$ of agents in the full domain, where each agent i has plan P_i from the relaxed domain. Then the *full domain* is a multiagent planning

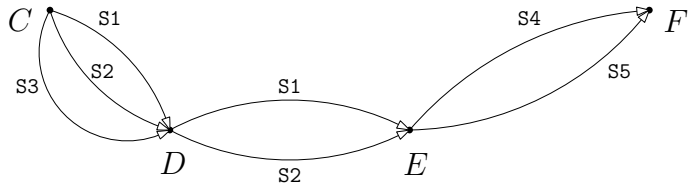


FIGURE 2. An example of the full domain with stops C, D, E and F for the merged plan of two single-agent plans $P = \{C \xrightarrow{\{1\}} D \xrightarrow{\{1,2\}} E \xrightarrow{\{1\}} F\}$

domain constructed using a *merged plan* P of single-agent plans $P_1, \ldots, P_n$ defined by formula

$$P = \bigcup_{i=1}^{n} P_i = (V', E', l')$$

where we interpret $\bigcup$ as the union of graphs that would result from interpreting each plan as a set of edges connecting stops. More specifically, given a set of single-agent plans, the plan merging operator $\bigcup$ computes its result in three steps: First, it transforms every single-agent plan P_i to a directed graph $T_i = (V_i, E_i)$ where the nodes V_i are the stops from the single-agent plan P_i and the edges E_i represent the atomic travel actions of P_i (for instance, a plan $P_1 = \langle C \to D, D \to E, E \to F \rangle$ is transformed to a directed graph $T_1 = \{C \to D \to E \to F\}$). Second, the merging operator performs a graph union operation $\bigcup_{i=1}^{n} T_i = (V', E', l')$ over the directed graphs and sets $V' = \bigcup_{i=1}^{n} V_i$, $E' = \bigcup_{i=1}^{n} E_i$, and labels every edge $e = (A, B) \in E'$ with the numbers of agents that are using the edge by a labelling function $l' : V' \times V' \to 2^S$. As an example , following Figure 1, the merged plan of plans of agent 1 travelling from C to F and sharing a journey from D to E with agent 2 would be computed as

$$\langle C \to D, D \to E, E \to F \rangle \cup \langle D \to E \rangle = \{C \xrightarrow{\{1\}} D \xrightarrow{\{1,2\}} E \xrightarrow{\{1\}} F\}$$

With this, the *full domain* is represented as a labelled directed multigraph $T' = (V', E_t, l, l')$ where the set of nodes V' represents the stops that are present in the merged plan P of plans from the relaxed domain. A set of edges E_t represents the journey services from the timetable. The labelling function $l : E_t \to \langle s, t_A, \tau \rangle$ returns a triple of a unique service name s, a departure time t_A from stop A, and a duration τ of the service journey between stops A and B for each edge $e = (A, B) \in E_t$. The labelling function $l' : V' \times V' \to 2^S$ labels every edge $e \in E'$ with the number of agents using it.

A joint plan π with a timetable is a sequence $\pi = \langle a^1 \ldots a^k \rangle$ of joint actions. Each joint action $a_j \in \pi$ represents a subset $S_j \subseteq S$ agents travelling together using a specific service s_j.

In the example of the full domain in Figure 2, the agents can travel using some subset of five different services `S1` to `S5`. The full domain example is based on the group of agent 1 (travelling from C to F) and agent 2 (travelling from D to E) where initial single-agent plans have been found in the relaxed domain shown in Figure 1. In order to travel from C to D using service `S1`, an agent must be present at stop C before the departure of service `S1` to D.

3.2. **Multiagent Planning Problem.** To model the ridesharing problem, we use a multiagent planning formalism which is based on MA-STRIPS [5] and coalition-planning games [6]. States are represented by sets of ground fluents, actions are tuples $a = \langle pre(a), \mathit{eff}(a)\rangle$. These fluents are logical propositions describing aspects of the current state that may change over time, e.g., $at(a_1, l_1)$ to express that agent a_1 is at location l_1. After the execution of action a, positive fluents p from $\mathit{eff}(a)$ are added to the state and negative fluents $\neg p$ are deleted from the state. For example, an action $travel(A, X, Y)$, when applied to the case of $A = a_1$ travelling from $X = l_1$ to $Y = l_2$ would make $at(a_1, l_1)$ false and $at(a_1, l_2)$ true. Each agent has individual goals and actions with associated costs. There is no extra reward for achieving the goal, the total utility received by an agent is simply the inverse of the cost incurred by the plan executed to achieve the goal. In the ridesharing domain, the agents are the travellers, located in their origin locations in the initial state, and attempting to achieve goal states where they are at their destination locations. Agents contains the representation of the initial and goal state, whereas journey planning for the agents is performed centrally.

More formally, following the notation of [29], a *multiagent planning problem* is a tuple

$$\Pi = \langle N, F, I, \{G_i\}_{i=1}^n, \{A_i\}_{i=1}^n, \Psi, \{c_i\}_{i=1}^n\rangle$$

where

- $N = \{1, \ldots, n\}$ is the set of agents,
- F is the set of fluents,
- $I \subseteq F$ is the initial state,
- $G_i \subseteq F$ is agent i's goal,
- A_i is agent i's action set,
- $\Psi : A \to \{0, 1\}$ is an admissibility function,
- $c_i : \times_{i=1}^n A_i \to \mathbb{R}$ is the cost function of agent i.

$A = A_1 \times \ldots \times A_n$ is the joint action set assuming a concurrent, synchronous execution model, and $G = \wedge_i G_i$ is the conjunction of all agents' individual goals. The assumption of synchronous action among agents here is an important simplification to make the problem more tractable. We will see below how it is possible to determine specific synchronisation points for jointly travelling agents when mapping the problem to the full timetabling information. A multiagent planning problem typically imposes concurrency constraints regarding actions that cannot or have to be performed concurrently by different agents to succeed which the authors of [29] encode using an admissibility function Ψ, with $\Psi(a) = 1$ if the joint action a is executable, and $\Psi(a) = 0$ otherwise.

A *plan* $\pi = \langle a^1, \ldots, a^k\rangle$ is a sequence of joint actions $a^j \in A$ such that a^1 is applicable in the initial state I (i.e., $pre(a^1) \subseteq I$), and a^j is applicable following the application of $a^1, \ldots, a^{j-1}$. We say that π *solves* the multiagent planning problem Π if the goal state G is satisfied following the application of all actions in π in sequence. The cost of a plan π to agent i is given by $C_i(\pi) = \sum_{j=1}^k c_i(a^j)$. Each agent's contribution to a plan π is denoted by π_i (a sequence of $a_i \in A_i$).

3.3. **Timetabled Transport Ridesharing Problem.** The real-world ridesharing domain used in this paper is based on the large and complex public transport network in the UK. An agent representing a passenger is able to use different means of transport during its journey: walking, trains, and coaches. The aim of each agent

is to get from its starting location to its final destination at the lowest possible cost. The cost of an agent's journey can be based on the weighted sum of several criteria such as journey duration, ticket price, mode of transport, and number of agents travelling together.

For the purposes of this paper, we will make the assumption that sharing a part of a journey with other agents is cheaper than travelling alone. While this may not currently hold in many public transport systems, defining hypothetical cost functions that reflect this would help assess the potential benefit of introducing such pricing schemes. This means that our cost functions reflect *synergies* occurring from the joint use of a resource, and this can be easily accommodated within the framework of best-response planning, where these positive effects on cost are simply treated as "negative contention", i.e., the cost to each agent when sharing a resource simply decreases instead of increasing. Note that this does not imply that every time an agent decreases her local cost this will benefit everybody else. For example, agent A might abandon the plan to share with B in order to reduce her overall cost, and join agent C instead, thus increasing B's cost, who will now travel alone. Thereupon B will try to improve on this result (and so on), the important property of BRP being that this process is guaranteed to terminate, and will result in a joint plan no individual agent can improve further on. Also, it is worth pointing out that joint plan will not necessarily be globally optimal – its quality will depend on the initial plan computed before the best-response process.

The ridesharing problem is then, for a given travel demand expressed as a set of origin-destination pairs, one for each agent, finding groups of agents and corresponding shared journey plans. We define the ridesharing problem more formally by presenting definitions for problem instances and our formal solution concept: A *timetabled transport ridesharing problem* is a triple $P = \langle T, T', G \rangle$, where

- $T = (V, E, w)$ is the *relaxed domain* containing a set V of *public transport stops*,
- $T' = (V', E_t, l, l')$ is the *full domain* over the subset $V' \subseteq V$ of public transport stops, and
- $G = \{(o_1, d_1), \ldots, (o_c, d_c)\}$ is a *set of agent trips* (an agent's goal is to travel from an origin to a destination), where each agent's trip $g \in G$ is represented by a tuple $g = (o, d)$ denoting the agent's origin $o \in V$ and destination $d \in V$.

A solution to this problem is a *joint plan* $\pi = \langle a^1, \ldots, a^k \rangle$ specifying fully the shared journeys of agents in terms of connections from the timetable and fulfilling all agent trips $g \in G$. From the many joint plans possible, we are looking for such a joint plan that correspond to a Nash equilibrium, i.e., where no agent/traveller can unilaterally improve its individual journey cost.

4. Ridesharing Planning Algorithm

Once we have formalised the problem, we can proceed to a detailed description of the ridesharing planning algorithm. The algorithm takes as an input the timetabled transport ridesharing problem $P = \langle T, T', G \rangle$, a maximum travel group size $n_{\max}$, and a maximum bearing difference $\Delta\varphi$. A bearing $\varphi(t)$ for a trip $t = (o, d)$ is defined as an angle in degrees, measured in the clockwise direction, between the north reference ray and the origin-destination ray. Bearing of a trip is used to identify trips with a similar direction as these are more suitable for ridesharing

than trips with opposite bearing. The output of the algorithm is a joint plan $\pi = \langle a^1, \ldots, a^k \rangle$ that fulfils all agent trips $g \in G$.

The main problem when planning for an identified group of agents with a centralised multiagent planner is the exponential blowup in the action space which is caused by using concurrent, independent actions [29]. Using a naive PDDL translation has proven that a direct application of a centralised multiagent planner to this problem does not scale well. As mentioned above, we tackle the complexity of the domain by breaking the planning process down into different phases that avoid dealing with the full fine-grained timetable data from the outset. The overall algorithm, which is shown in Figure 3, is designed to work in four phases, which we will now describe in detail.

4.1. **The Trip Grouping Phase.** The algorithm starts with the *trip grouping phase* where the trips $G = \{(o_1, d_1), \ldots, (o_c, d_c)\}$ are grouped into groups of at most $n_{\max}$ agents. Groups are created incrementally from G, until G becomes empty, in the following way: First, pick a trip $g' \in G$ at random. Then, create a set of candidate trips $G' = \{g \in G | \mathrm{bd}(g, g') \leq \Delta\varphi\}$ that have a similar bearing as g' (function $\mathrm{bd}(g, g')$ calculates the bearing difference between trips g and g'). Next, create a group $G_j \subseteq G'$ by selecting at most $n_{\max}$ trips with minimum spatial difference $\mathrm{sd}(\cdot, g')$ to g'. Here, the spatial difference $\mathrm{sd}(g, g')$ of two trips g and g' is defined as

$$\mathrm{sd}(g, g') = |o, o'| + |d, d'|,$$

where $|o, o'|$ denotes the direct distance between the origins of the two trips, and $|d, d'|$ the direct distance between their destinations. Once a group G_j is created, the trips $g \in G_j$ are deleted from the set of all trips G.

For each group G_j, a joint journey plan π with a timetable is found by applying the next three phases of the algorithm.

4.2. **The Trip Planning Phase.** In the *trip planning phase*, an initial journey is found for each agent i from the set of agents G_j using the relaxed domain $T = (V, E, w)$ where the action set is identical for every agent and contains all transport services available in the transport network. A journey for each agent is calculated independently of other agents in the scenario using a single-agent planner. As a result, each agent is assigned a single-agent plan P_i which will be further optimised in the next phase. This approach makes sense in our domain because the agents do not need each other to achieve their goals and they cannot invalidate each other's plans. A PDDL specification for the relaxed domain is shown in section 4.5.2.

4.3. **The Best-response Phase.** In the *best-response phase*, which is also based on the relaxed domain. Again, the action set is identical for every agent and contains all transport services available in the transport network. The algorithm uses the best-response planning algorithm as described below. It iteratively creates and solves simpler best-response planning problems from the point of view of each individual agent. In the case of the relaxed domain, the best-response planning problem looks almost the same as a problem of finding a single-agent journey. The difference is that, as we have explained in section 3.3, we make the assumption that the cost of travelling is smaller when an agent uses a connection which is used by one or more other agents. A specific cost function used for the evaluation of the algorithm is defined in section 5.2.

Input

- Timetabled transport ridesharing problem $P = \langle T, T', G \rangle$
- Maximum travel group size $n_{\max}$
- Maximum bearing difference $\Delta\varphi$

1. The trip grouping phase

Set $j = 0$

While $G \neq \emptyset$ **do**

(1) Pick a trip $g' \in G$ at random
(2) Create a set of candidate trips $G' = \{g \in G | \mathrm{bd}(g, g') \leq \Delta\varphi\}$
(3) Create a group $G_j \subseteq G'$ by selecting at most $n_{\max}$ trips with minimum spatial difference $\mathrm{sd}(\cdot, g')$ to g'
(4) Delete trips $t \in G_j$ from G
(5) Set $j = j + 1$

For each created group $G_j = \{1, \ldots, n\}$ **do** the next three phases

2. The trip planning phase

For $i = 1, \ldots, n$ **do**

Find an initial journey for agent i using a single-agent planner

3. The best-response phase

Do until no change in the cost of the joint plan

For $i = 1, \ldots, n$ **do**

(1) Create a simpler best-response planning problem from the point of view of agent i
(2) Minimise the cost of i's plan without changing the plans of others

4. The timetabling phase

Identify independent groups of agents $I = \{1, \ldots, m\}$

For $i = 1, \ldots, m$ **do**

(1) Find the relevant timetable for group i
(2) Match the joint plan of i to timetable using a temporal single-agent planner in the full domain with the relevant timetable

Output

- Joint plan $\pi = \langle a^1, \ldots, a^k \rangle$ that fulfils all agent trips $g \in G$

FIGURE 3. Four-phase algorithm for finding shared journeys for agents

Iterations over agents continue until there is no change in the cost of the joint plan between two successive iterations. This means that the joint plan cannot be further improved using the best-response approach. The purpose of this is not only to exploit local, "greedy" optimisations for single agents in an overall schedule of plans. It also ensures that the proposed joint solution is compatible

with the incentives of individual agents, i.e., they could do no better on their own by deviating from it. The fact that the first iteration of the best-response optimisation starts from initial plans that agents can perform on their own ensures this (any subsequent plan generated will be cheaper to them). The output of the best-response phase is a merged plan P of the single-agent plans in the relaxed domain (defined in section 3.1) that specifies which connections the agents use for their journeys and which segments of their journeys are shared. The merged plan P will be matched to the timetable in the final phase of the algorithm.

4.3.1. *Best-response Planning.* The *best-response planning* algorithm proposed in [29] is an algorithm which, given a solution π^k to a multiagent planning problem Π, finds a solution π^{k+1} to a *transformed planning problem* Π_i with minimum cost $C_i(\pi^{k+1})$ for agent i among all possible solutions, while considering all other agents' plans to be fixed:

$$\pi^{k+1} = \arg\min\{C_i(\pi) | \pi \text{ identical to } \pi^k \text{ for all } j \neq i\}$$

The transformed planning problem Π_i is obtained by rewriting the original problem Π so that all other agents' actions are fixed, and agent i can only choose its own actions in such a way that all other agents still can perform their original actions. Since Π_i is a single-agent planning problem, any cost-optimal planner can be used as a best-response planner.

In [29], the authors show how for a class of congestion planning problems, where all fluents are *private*, the transformation they propose allows the algorithm to converge to a Nash equilibrium if agents iteratively perform best-response steps using an optimal planner. This requires that every agent can perform its actions without requiring another agent, and hence can achieve its goal in principle on its own, and conversely, that no agent can invalidate other agents' plans. Assuming infinite capacity of vehicles (or, more realistically, large enough capacities to accommodate at least the number of agents for whom we are trying to find a plan), the relaxed domain is an instance of a congestion planning problem: following the definition of a congestion planning problem in [29], all actions are private, as every agent can use a means of transport on their own and the other agents' concurrently taken actions only affect action cost. The convergence of the best-response phase derives from the theorem presented in [29] which states that for any congestion planning problem, best-response planning converges to a pure-strategy Nash equilibrium.

The best-response planner works in two phases: In the first phase, an initial plan for each agent is computed (e.g., each agent plans independently or a centralised multiagent planner is used). In the second phase, the planner solves simpler best-response planning problems from the point of view of each individual agent. The goal of the planner in a best-response planning problem is to minimise the cost of an agent's plan without changing the plans of others (though the cost of their plans might change as explained in section 3.3). Consequently, it optimises a plan of each agent with respect to the current joint plan.

This approach has several advantages. It supports full concurrency of actions and the best-response phase avoids the exponential blowup in the action space resulting in much improved scalability. For the class of potential games [32], it guarantees convergence to a Nash equilibrium. On the other hand, it does not guarantee the optimality of a solution, i.e., the quality of the equilibrium in terms of overall efficiency is not guaranteed (it depends on which initial plan the agents

start off with). However, experiments have proven that it can be successfully used for improving general multiagent plans [29].

4.4. **The Timetabling Phase.** In the final *timetabling phase*, the optimised shared journeys are matched against timetables using a temporal single-agent planner which assumes the full domain. For this, in a first step, independent groups of agents with respect to journey sharing are identified. An independent group of agents is defined as an edge disjoint subgraph of the merged plan P. This means that actions of independent groups do not affect each other so it is possible to find a timetable for each independent group separately.

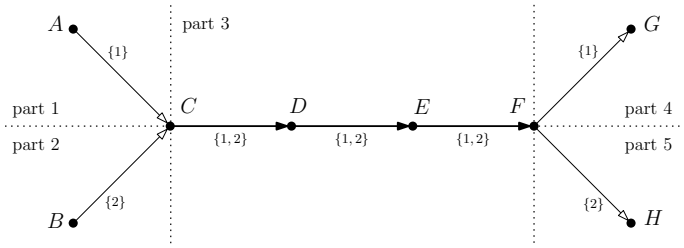


FIGURE 4. Parts of the group journey of two agents

Then, for every independent group, *parts* of the group journey are identified. A *part* of the group journey is defined as a maximal continuous segment of the group journey which is performed by the same set of agents. As an example, there is a group of two agents that share a segment of their journeys in Figure 4: Agent 1 travels from A to G while agent 2 travels from B to H. Their group journey has five parts, with the shared part (part 3) of their journey occurring between stops C and F.

In order to use both direct and stopping trains when the group journey is matched to the timetable, the *relevant timetable* for a group journey is composed in the following way: for every part of the group journey, return all timetable services in the direction of agents' journeys which connect the stops in that part. An example of the relevant timetable for a group of agents from the previous example is shown in Figure 5. Now, the agents can travel using the direct train `T1` or using train `T2` with intermediate stops.

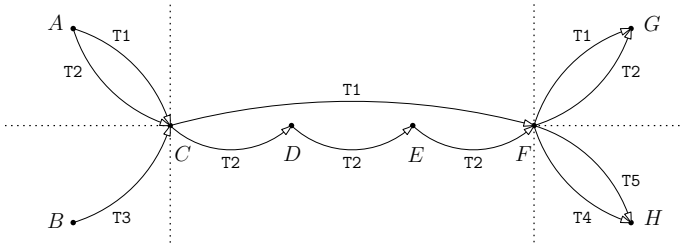


FIGURE 5. The full domain with services from the relevant timetable. There are five different trains `T1` to `T5`, and train `T1` is a direct train.

The relevant timetable for the group journey is used with the aim to cut down the amount of data that will be given to a temporal single-agent planner. For instance, there are 9,881 timetabled connections for trains in the Yorkshire area. For an example journey of 4 agents, there are only 634 services in the relevant timetable which is approximately 6% of the data. As a result, the temporal single-agent planner gets only the necessary amount of data as input, to prevent the time-consuming exploration of irrelevant regions of the state space.

In the timetabling phase, every agent in a group of agents tries to spend the shortest possible time on its journey. When matching the plan to the timetable, the temporal planner tries to minimise the sum of durations of agents' journeys including waiting times between services. A PDDL specification for the full domain is shown in section 4.5.2.

Once the timetabling phase finishes, the algorithm adds a joint plan $\pi^j = \langle a^1, \ldots, a^k \rangle$ for the identified group of agents $G_j \in G$ to the final joint plan $\pi = \pi \bigcup \pi^j$. The algorithm then proceeds to the next group $G_j \in G$.

4.5. **Implementation.** This section describes two important aspects of the algorithm implementation. It deals with the conversion of public transport timetables data to the Planning Domain Definition Language and with the choice of planners for implementing the individual phases of the algorithm.

4.5.1. *Importing Timetables.* To be able to use timetables data of public transport services (cf. section 5.1) with modern AI planning systems, it has to be converted to the Planning Domain Definition Language (PDDL). We transformed the data in three subsequent stages. First, we transformed the NPTDR and NaPTAN XML data to a spatially-enabled PostgreSQL database. Second, we automatically processed and optimised the data in the database. The data processing by SQL functions in the procedural PL/pgSQL language included the following steps: merging bus bays at bus stations and parts of train stations, introducing walking connections to enable multi-modal journeys, and eliminating duplicates from the timetable. Finally, we created a script for generating PDDL specifications based on the data in the database. More details about the data processing and PDDL specifications can be found in [26].

4.5.2. *PDDL definitions.* In the relaxed domain used in the trip planning and best-response phase, a single agent aims to travel from its origin to its destination. The domain file contains two predicates, two functions and only one action, cf. Figure 7. The predicate `connection` is true when there is an edge from `?origin` to `?destination` (there are separate edges for walking, travel by bus or train), the predicate `at` denotes the current location of the agent. The function `time` returns the cost of travelling from the location `?origin` to `?destination`. The action `go` moves the agent from the location `?o` to `?d` and it increases the total cost of the plan which is stored by the `total-cost` function. The problem file then contains origin and destination of the agent, a list of stops, and a list of connections between the stops and their costs.

In the full domain used in the timetabling phase, multiple agents aim to travel from their origins to their destinations. The domain is based on the result of merging individual plans P from the relaxed domain. Therefore, it contains only the stops that are present in the union of these plans, with the shared parts of the journeys and "who shares which part of the journey" already specified. In the

process of finding a plan for the full domain, a joint plan of the group of agents is instantiated with concrete timetabled services.

```
(define (domain travelplanner)
    (:requirements :typing :action-costs)
    (:types location)
    (:predicates
        (connection ?origin - location ?destination - location)
        (at ?loc - location)
    )
    (:functions
        (time ?origin - location ?destination - location)
        (total-cost)
    )
    (:action go
        :parameters (?o ?d - location)
        :precondition (and (at ?o) (connection ?o ?d) )
        :effect (and
            (at ?d) (not (at ?o))
            (increase (total-cost) (time ?o ?d)) )
    )
)
```

FIGURE 6. The domain file for the relaxed domain

```
(:durative-action go-agent-1-2_A-B
 :parameters (?s - service)
 :duration (= ?duration (+
     (- (+ (departure A B ?s) (runtime A B ?s)) (agent-time agent1))
     (- (+ (departure A B ?s) (runtime A B ?s)) (agent-time agent2))
 ))
 :condition (and
     (at start (connection A B ?s))
     (at start (at agent1 A))
     (at start (<= (agent-time agent1) (departure A B ?s)))
     (at start (at agent2 A))
     (at start (<= (agent-time agent2) (departure A B ?s)))
 )
 :effect (and
     (at end (at agent1 B))
     (at start (not (at agent1 A)))
     (at end (assign (agent-time agent1)
                     (+ (departure A B ?s) (runtime A B ?s))))
     (at end (at agent2 B))
     (at start (not (at agent2 A)))
     (at end (assign (agent-time agent2)
                     (+ (departure A B ?s) (runtime A B ?s))))
))
```

FIGURE 7. A durative action *go-agent-1-2_A-B* in the domain file for the full domain

The domain file contains a list of partially instantiated durative actions for travelling from one stop to another, where origin, destination, and agents using this action are instantiated, and the only free variable is the name of the service the agents are going to use. The function `(agent-time ?a - agent)` is used to store

the current time of the agent `?a`. An example of a durative action is shown in Figure 6. The durative action *go-agent-1-2_A-B* enables agent 1 and 2 to travel together from stop A to stop B. If the travel from A to B is shared by three agents, the domain file would contain an action *go-agent-1-2-3_A-B*.

Let N be the number of agents travelling together, at_i the current time of agent i, $d_{AB}(s)$ the departure time of service s from the stop A to B and $r_{AB}(s)$ its duration. Then, the duration D_{AB} of the action to travel from the stop A to B is computed as $D_{AB} = \sum_{i=1}^{N} (d_{AB}(s) + r_{AB}(s) - at_i)$. The temporal planner tries to minimise the sum of the durations of agents' journeys. In other words, it tries to find a journey with minimal waiting times between services.

The conditions of the action are the following: there must be a connection by the service s between the stops A, B and the agents must be present at the stop A before the departure of service s. Once the action is executed, the agents are located at stop B and their current time is set to the arrival of the service s at stop B. The problem file contains origins and destinations of the agents and a list of services and their departures and durations.

4.5.3. *Planners.* All three single-agent planners used for the implementation were taken from recent International Planning Competitions from 2008 and 2011. We use LAMA [37] in the trip planning and the best-response phase, a sequential *satisficing* (as opposed to cost-optimal) planner which searches for any plan that solves a given problem and does not guarantee optimality of the plans computed. LAMA is a propositional planning system based on heuristic state-space search. Its core feature is the usage of landmarks, i.e., propositions that must be true in every solution of a planning problem.

SGPlan_6 [28] and POPF2 [9] are the two temporal satisficing planners used in the timetabling phase. Such temporal planners take the duration of actions into account and try to minimise makespan (i.e., total duration) of a plan but do not guarantee optimality. The two planners use different search strategies and usually produce different results. This allows us to run them in sequence on every problem and to pick the plan with the shortest duration. It is not strictly necessary to run both planners, one could save computation effort by trusting one of them.

In many of the experiments, the SGPlan_6 and POPF2 used in the timetabling phase returned some plans in the first minute but then they continued exploration of the search space without returning any better plan. To account for this, we imposed a time limit for each planner in the temporal planning stage to 2 minutes for a group of up to 4 agents and 4 minutes otherwise.

5. Evaluation

We have evaluated the proposed ridesharing algorithm on realistic scenarios based on real-world public transport timetables and travel demand data for the Yorkshire area of the United Kingdom. The size of the area was dictated solely by our need to evaluate the algorithm on the whole travel demand (approximately 100,000 train trips per day). The ridesharing planning algorithm itself scales up well up to the area of the whole UK, as was shown in our previous work [27]. However, there the algorithm was evaluated on travel demand that was very sparsely and randomly sampled, not necessarily showing any correlation to actual travel demand profiles.

TABLE 1. Numbers of trips per day in the Yorkshire area [21, 22]

Transport mode	**Modal split**	**100% trips**	**50% trips**	**5% trips**
Trains	5.3%	106,035	53,017	5,302
Coaches	0.3%	6,002	3,001	300
Local buses	6.0%	120,039	60,020	6,002
Passenger cars	88.3%	1,766,576	883,288	88,329
Total	100.0%	1,998,651	999,326	99,933

5.1. **Domain Data.** The timetables of public transport services were taken from the National Public Transport Data Repository (NPTDR, data.gov.uk/dataset/nptdr) which is publicly available from the Department for Transport of the British Government. For the evaluation of the algorithm, we used data from 2010, which is provided in TransXChange XML, in an XML-based UK standard for interchange of route and timetable data.

National Public Transport Access Nodes (NaPTAN, data.gov.uk/dataset/naptan) is a UK national system for uniquely identifying all the points of access to public transport. Every point of access (bus stop, railway station, etc.) is identified by an ATCO code (a unique identifier for all points of access to public transport in the UK), e.g., *9100YORK* for York Rail Station. Each stop in the NaPTAN XML data is also supplemented by common name, latitude, longitude, address and other pieces of information. This data also contains information about how the stops are grouped together (e.g., several bus bays that are located at the same bus station).

The experiments are situated in the Yorkshire area (East and West Yorkshire, East Riding of Yorkshire, York, and Selby administrative areas) which covers an area of approximately 130 by 70 km, i.e., around 9,100 km^2. According to the UK origin-destination census data from 2001 [22], there are 2 million passenger trips a day in the Yorkshire area. In order to focus on the timetabled public transport trips, the modal split in the UK across different modes of transport in 2001 [21] was used to estimate the number of trips for each mode, cf. Table 1.

Since we assume that all agents are travelling on the same day and that all journeys must be completed within 24 hours, in what follows below we consider only public transport timetables data for Tuesdays (this is an arbitrary choice that could be changed without any problem).

5.2. **Cost Model.** The timetable data used in this paper (cf. previous section) contains neither information about ticket prices nor distances between adjacent stops, only durations of journeys from one stop to another. This significantly restricts the design of a cost functions used for the planning problems. Therefore, the cost functions used in this paper are based solely on the duration of journeys. The cost $c_{i,n}$ for agent i travelling from A to B in a group of n agents is then defined by equation (5.1):

$$c_{i,n} = \left(\tfrac{1}{n}\,0.8 + 0.2\right) c_i \tag{5.1}$$

where c_i is the individual cost of the single action to i when travelling alone. In this paper, we take this to be equal to the duration of the journey from A to B.

This is designed to approximately model the discount for the passengers if they buy a group ticket: The more agents travel together, the cheaper the shared (leg of a) journey becomes for each agent. Also, an agent cannot travel any cheaper

TABLE 2. Experiment scenarios parameters overview

Scenario parameter	**Parameter values**
Travel demand generation	{realistic, random}
Ridesharing demand proportion	{100%, 50%, 5%}
Modes of transport	{trains only, trains and coaches}
Maximum travel group size	$\{2, 4, 6, 8\}$

than 20% of the single-agent cost. In reality, pricing for group tickets could vary, and while our experimental results assume this specific setup, the actual price calculation could be easily replaced by any alternative model.

5.3. **Experiment Scenarios.** We used the following parameters as factors in experiment scenarios: (1) travel demand generation; (2) ridesharing demand proportion; (3) modes of transport considered; (4) maximum travel group size. The values of the parameters are summarised in Table 2. We set the maximum bearing difference parameter of the algorithm to $\Delta\varphi = 25$ degrees for all scenarios.

Travel Demand Generation. We use two types of travel demand. The *realistic* travel demand generation is based on the UK census 2001 origin-destination data [22] that contains numbers of trips carried out from every origin district to every other destination district. District-to-district trip counts are mapped to stop-to-stop trip counts in the following way: For each origin-destination district pair, the desired number of trips is generated randomly from the Cartesian product of stops in the origin and destination district. Since the UK census origin-destination data is not provided at the level of granularity required to select concrete stops in the travel network, we have sampled these within each district with probability proportional to the density of services passing through a stop. This is based on the assumption that service density roughly follows numbers of travellers using a stop. In addition to the realistic demand, we also experimented using *random* travel demand generated randomly from the Cartesian product of stops in the Yorkshire area, assuming a uniform distribution over stops.

From the travel demand distribution generated, only trips with a straight-line distance between the origin and the destination in the interval 25–100 km are used for the evaluation (when using roads or rail tracks, this interval stretches approximately to a real distance of 40–160 km). This interval was chosen to filter out trips that are too short to be planned in advance and therefore not very suitable for sharing. This led to the removal of 86% of all trips in the realistic travel demand generation process and to the removal of 30% of all trips in the random travel demand generation process.

Ridesharing Demand Proportion. In order to observe the behaviour of the system with different densities of trips we set the portion of travel demand to 100%, 50%, and 5% of the total number of trips.

Modes of Transport. In order to evaluate the behaviour of the algorithm on both a unimodal and a multi-modal public transport network, *trains only* and a combination of *trains and coaches* were used in the experiments. In the Yorkshire area, there are 150 (201) stops, 330 (495) connections in the relaxed domain, and 9,881 (10,289) connections in the timetable for trains (and coaches).

Maximum Travel Group Size. The maximum travel group size $n_{\max}$ is one of the algorithm's inputs that restricts the size of groups created in the trip grouping phase. We set this parameter to 2, 4, 6, and 8 as after initial testing, it became clear that groups of a larger size are almost never practicable.

5.4. **Metrics.** We evaluate the performance of the algorithm in terms of three different metrics: improvement in the cost of agents' journeys, their prolongation, and the computation time of the algorithm.

Cost Improvement. To evaluate the net benefit of using our method for ridesharing, we calculate the cost improvement for the agents' journeys. To calculate this, recalling that $C_i(\pi) = \sum_j c_i(a^j)$ for a plan is the cost of a plan $\pi = \langle a^1, \ldots, a^k \rangle$ to agent i, assume $n(a^j)$ returns the number of agents with whom the jth step of the plan is shared. We can define the cost of a shared travel plan $C_i^{'}(\pi) = \sum_j c_{i,n(a^j)}(a^j)$ using equation (5.1). With this, we can calculate the cost improvement ΔC as follows:

$$\Delta C = \frac{\sum_{i \in N} C_i(\pi_i) - \sum_{i \in N} C_i^{'}(\pi_N)}{\sum_{i \in N} C_i(\pi_i)} \tag{5.2}$$

where N is the set of all agents, π_i is the single-agent plan initially computed for agent i, and π_N is the final joint plan of all agents after completion of the algorithm (which, though it is in reality a set of several plans for different subgroups of N, is interpreted as a single plan for the "grand coalition" N and reflects how subgroups within N share parts of their individual journeys).

Prolongation. On the one hand, ridesharing is beneficial in terms of cost. On the other hand, a shared journey has a longer duration than a single-agent journey in most cases, because agents have to take later services than they could use on their own if they are waiting for co-travellers to arrive. In order to evaluate this trade-off, we measure journey prolongation. Assume that $T_{\mathrm{i}}(\pi)$ is the total duration of a plan to agent i in plan π, and, as above, π_i/π_N denote the initial single-agent plans and the shared joint plan at the end of the timetabling phase, respectively. Then, the prolongation ΔT of a journey is defined as follows:

$$\Delta T = \frac{\sum_{i \in N} T_i(\pi_N) - \sum_{i \in N} T_i(\pi_i)}{\sum_{i \in N} T_i(\pi_i)} \tag{5.3}$$

Computation Time. To assess the scalability of the algorithm, we measure the amount of time needed to create groups of agents in the first phase of the algorithm and then to plan shared journeys for all agents in each group.

5.5. **Results.** In this section, we present the results of the evaluation in terms of journey cost improvement, journey prolongation, and computation time of the algorithm. Exhaustive experiment design was used; all metrics were evaluated for all combinations of all values of all scenario parameters. Specifically, for each type of travel demand generation, we tested all combinations of modes of transport, and for each ridesharing demand proportion, we generated the travel demand. Then for each maximum group size, the whole travel demand is an input for the algorithm which in its trip grouping phase creates the groups of agents for ridesharing. From the set of all groups created, a detailed journey plan with a timetable is found in the last three phases of the algorithm for a sample of 80 randomly chosen groups (sampling was performed to reduce experiment computational time while maintaining

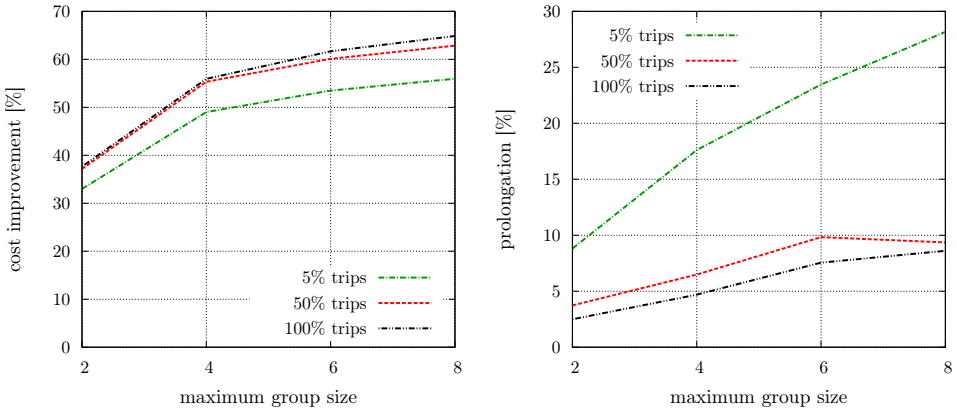


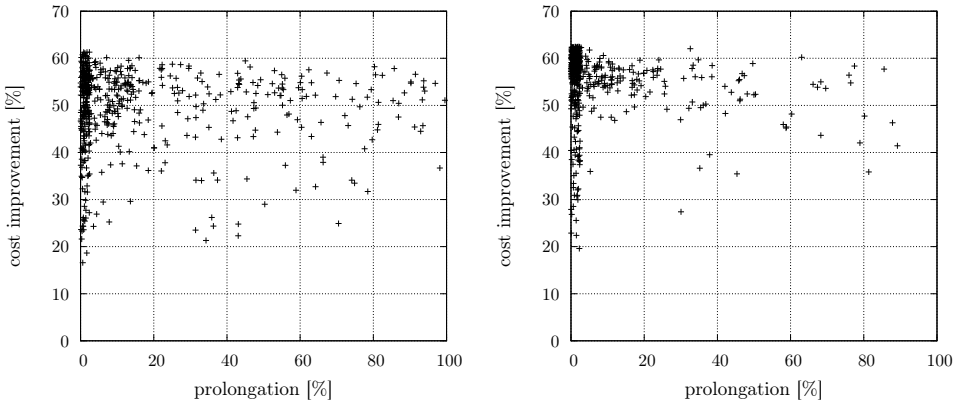
FIGURE 8. Average cost improvement and prolongation against maximum group size (realistic travel demand, trains and coaches)

significance of the results). Each possible experiment configuration is averaged over 8 stochastic travel demand generation instances. This leads to an overall number of 30,720 groups of agents over which the algorithm was evaluated. The results obtained are based on running the algorithm on one core of a 3.2 GHz Intel Core i7 processor of a Linux desktop computer with a PostgreSQL 9.1 database (spatially enabled with PostGIS 2.0.1).

Cost Improvement. The average cost improvement obtained in our experiments is shown in Figure 8. It shows that the more agents are grouped together in the trip grouping phase of the algorithm, the higher the improvement. These results were obtained based on the specific cost function (5.1) we have introduced to favour ridesharing, and which would have to be adapted to the specific cost structure that is present in a given transport system. Also, the extent to which longer journey times are acceptable for the traveller depends on their preferences, but these could be easily adapted by using different cost functions.

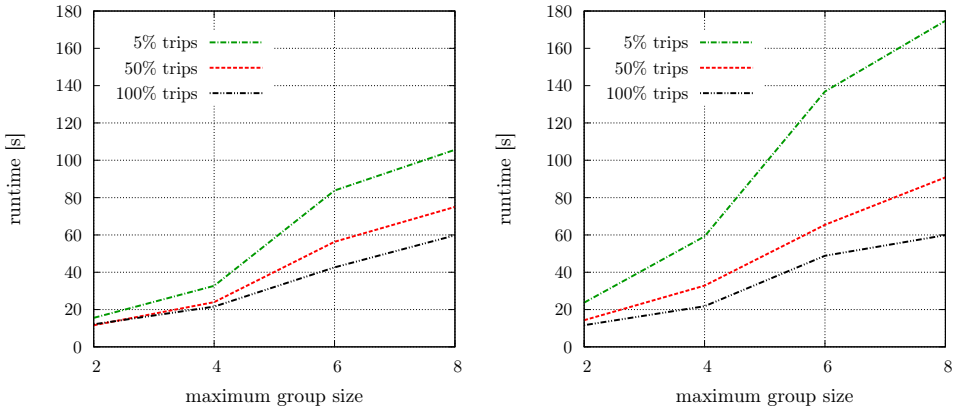
Prolongation. The average prolongation of journeys is shown in Figure 8 where 8% of groups with prolongation greater than 100% is filtered out from the average calculation (these are the journeys which, though feasible, are unlikely to be accepted by travellers). The graph shows that the more agents are grouped together in the trip grouping phase of the algorithm, the higher the prolongation. Furthermore, the prolongation with the 5% ridesharing proportion is much higher then when considering 50% or 100% ridesharing proportion. As the density of trips drops, the agents in groups are more spatially dispersed, which causes higher relative prolongation ratios.

Figure 9 shows a scatter plot of cost improvement versus prolongation for individual trips for 5% and 50% ridesharing proportion. It can be observed that with a higher ridesharing proportion, the majority of the groups has either prolongation very close to 0% (identical trips are shared) or has a very high cost improvement (between 50% and 60%). With a lower ridesharing proportion, there are many more groups with lower cost improvement or higher prolongation. What is encouraging is that even for small populations of potential ridesharers, there are many shared



(A) 5% trips (B) 50% trips

FIGURE 9. Cost improvement against prolongation (realistic travel demand, trains and coaches, maximum group size $n_{\max} = 4$)



(A) Realistic travel demand (B) Random travel demand

FIGURE 10. Computation time against maximum group size (trains and coaches)

journeys with a good cost improvement and a reasonable prolongation. In our algorithm, the balance between the two criteria could be calibrated by changing the weights in the cost function.

Computation Time. The first graph in Figure 10 shows the overall computation times of the algorithm for one created group of agents from the realistic demand and a combination of trains and coaches. The trip grouping phase of the algorithm is very fast (200 ms per group on average). The algorithm spends the majority of the computation time solving the problem of finding a joint plan for the group of agents. The graph indicates that the overall computation time grows roughly linearly with increasing numbers of agents in a group, which confirms that the algorithm avoids the exponential blowup in the action space characteristic for centralised multiagent

planning. This is mainly a consequence of the best-response planning algorithm, and an expected result.

The second graph in Figure 10 shows the overall computation times of the algorithm for one created group of agents from the random demand and for a combination of trains and coaches. It can be observed that the algorithm is faster at the realistic 5% ridesharing proportion than for random trips. At the 50% and 100% ridesharing proportion, there is not a very big difference between the computation times. This suggests that the trips from the realistic 5% ridesharing proportion reflects the public transport network making the journey planning easier whereas it is harder to plan for trips distributed randomly.

Regarding the modes of transport in the scenario, it is harder to find joint plans when a combination of trains and coaches is considered (on average, runtimes are 25% higher for scenarios with trains and coaches). Considering a combination of trains and coaches does not significantly affect neither the cost improvement, nor the prolongation.

While the overall computation times are considerable (up to 2 minutes for a group of 8 agents from the realistic 5% ridesharing proportion), we should emphasise that the algorithm is effectively computing equilibrium solutions in multi-player games with hundreds of thousands of states. Considering this, the linear growth hints at having achieved a level of scalability based on the structure of the domain that is far above naive approaches to plan jointly in such state spaces.

Finally, we have evaluated the overall computation time for all trips from the travel demand. We were able to compute shared journey plans for approximately 13,500 trips from realistic 100% ridesharing proportion when considering a combination of trains and coaches. It took less than 75 minutes for each setting of maximum group size while using 8 cores of 3.2 GHz Intel Core i7 processor on three computers in parallel.

The ridesharing algorithm can be further parallelised down to a level of individual groups, bringing the computation time to few minutes for the whole demand. This follows from the structure of the algorithm: In the trip planning phase, the identification of initial single-agent plans for the travellers consists of a set of completely independent problems. In the best-response and timetabling phase, the planning problem of each group of agents is also completely independent from those of other groups or individual travellers.

6. Discussion

Our proposed algorithm clearly improves the cost of agents' journeys by sharing parts of the journeys, even though there is an inherent trade-off between cost improvement and the prolongation of journeys. On the one hand, the bigger the group, the better the improvement. On the other hand, the more agents share a journey, the higher the prolongation is likely to be. This will most likely lead to results that are not acceptable for users in larger groups. Whether prolongation or cost savings are more important in a given scenario will depend on the real preferences of travellers, and our system would allow them to customise these settings per individual on, for example, a Web-based ridesharing planner that would use our algorithm. It is also important to point out that our framework can be used without any significant modifications for any other cost function as appropriate

for the transport system in question, and, subject to availability of the required real-world timetabled transport data, for any other geographical region.

Next, note that trip planning and best-response phases of the algorithm are completely domain-independent and can therefore easily be used for other types of transport problems, e.g., to plan routes that avoid traffic jams or to schedule parcel deliveries. What is more, additional constraints such as staying at a location for some time or travelling together with a specific person can be easily accommodated within standard planning languages, and the use of standard planning technology also implies that our method will directly benefit from future improvements in planning algorithms. On the other hand, the trip grouping and the timetabling phase of the algorithm are domain-specific, providing an example of the specific design choices that have to be made from an engineering point of view when applying standard AI methods to problems of decentralised decision-making in transport.

From an algorithm and systems engineering perspective, using off-the-shelf problem solvers such as AI planning systems for a complex real-world domain like trip sharing brings an additional benefit, which is that we do not need to engineer novel optimisation algorithms for the combinatorial problems arising in this family of problems *from scratch*. While it is certainly possible that faster algorithms that produce better solutions may exist for specific problems, our approach enables us to formalise different types of similar problems with comparatively little effort and to make use of the best available search heuristics in a lightweight fashion. We believe that this is an effective way of developing decentralised resource allocation and process optimisation systems in domains like transport, where it can be expensive to develop a custom solution for every different class of problems although many of them share many common characteristics.

The presented experiments work with a demand for a *whole day* and therefore the generated joint plans are not restricted to any particular part of a day. In reality, however, travellers may not be so flexible in terms of timing their journeys. This problem can be easily solved by considering time constraints in the clustering performed in the trip grouping phase of the algorithm. Trips might e.g. be put into one group only if their preferred departure and/or arrival times do not differ by more than a given time difference. The performance achieved for lower trip densities corresponding to 5% ridesharing proportion suggests that attractive shared journeys would be found even for the maximum time difference of one hour (one hour constitutes approximately 4% of a day), or even less during peek hours when demand is more concentrated.

There are many potential uses of the proposed approach to real-world ridesharing: From a traveller's perspective, it can be used to exploit current ticket discounts for group travel while enjoying the company of friends, fellow workers, and other co-travellers. A web- or smartphone-based application can be built which would collect user preferences and constraints and propose shared journey plans. Further, in future applications our approach could be combined with the use of private cars to mix public and private modes of transport. This can be achieved with fairly small modifications. It would work in a similar way as walking is combined with public transport in our evaluation, except that car travel enables non-timetabled transport for more than one individual. This is an important extension, as in most realistic settings, successful ridesharing would certainly include private cars. In fact, without this, we are only considering a very hard problem, were travellers

have very limited flexibility. Naturally, if at least one person in each group has a car, this opens up (orders of magnitude) more options for joint trips. Also, for car sharing the cost benefit is arguably much higher, and can be much more objectively calculated than what we have assumed in our hypothetical cost function (e.g., cost/km divided by number of car passengers).

From a public policy and transport planning perspective, stakeholders in the public transport domain could use our method to predict customer behaviour when considering modifications to timetables, the introduction of new services, and modifications to pricing schemes to optimise usage, environmental footprint, and business revenue. Such scenario analysis could easily accommodate taking further factors into account, such as waiting times, travel interruptions for business and leisure activities, preferences of individuals to share trips with particular co-travellers, etc. In particular, it could give rise to new incentive schemes for ridesharing, such as discounts for group travel that depend on the cumulative amount of sharing or occupancy ratios along different legs of joint journeys involving various means of transport and changing groups of jointly travelling individuals.

7. Conclusion

We have presented a multiagent planning algorithm which is able to plan meaningful shared journeys using timetabled public transport services. The algorithm has been implemented and evaluated on realistic scenarios based on real-world UK transport data. Experiments with realistic travel demand show that, for a wide range of scenarios, the algorithm is capable of finding shared journeys with very attractive trade-offs between cost saving and journey time prolongation.

The algorithm exhibits very good scalability, scaling linearly with the number of trips processed, regardless of the size of travel groups considered. The algorithm is also amenable to massive parallelisation which can bring the time required for planning shared journeys for real-world travel demand down to minutes.

Finally, the cost of travel and flexibility of ridesharing can be significantly improved by sharing private cars. In the future, we plan to extend the algorithm towards multi-modal ridesharing in which groups of travellers can seamlessly transfer between timetabled and non-timetabled transport modes.

8. Acknowledgments

This work was supported by the EC within the framework of the SUPERHUB Project (grant agreement No. 289067) and by the Ministry of Education, Youth and Sports of Czech Republic (grant No. LD12044).

References

1. Gerardo Berbeglia, Jean-Franois Cordeau, and Gilbert Laporte, *Dynamic pickup and delivery problems.*, European Journal of Operational Research **202** (2010), no. 1, 8–15.
2. C. Boutilier and R. Brafman, *Partial-order planning with concurrent interacting actions*, Journal of Artificial Intelligence Research **14** (2001), 105–136.
3. M. Bowling, R. Jensen, and M. Veloso, *A formalization of equilibria for multiagent planning*, AAAI Workshop on Planning with and for Multiagent Systems, July 2002.
4. R. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz, *Planning Games*, Proceedings of the International Joint Conference on Artificial Intelligence, vol. 21, 2009, pp. 73–78.
5. Ronen I. Brafman and Carmel Domshlak, *From One to Many: Planning for Loosely Coupled Multi-Agent Systems*, Procs. ICAPS 2008, AAAI Press, 2008, pp. 28–35.

6. Ronen I. Brafman, Carmel Domshlak, Yagil Engel, and Moshe Tennenholtz, *Planning Games*, Procs. IJCAI 2009, July 2009, pp. 73–78.
7. Gerth S. Brodal and Riko Jacob, *Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries*, Electronic Notes in Theoretical Computer Science **92** (2004), no. 0, 3–15.
8. P. C. Buzing, A. W. ter Mors, J. M. Valk, and C. Witteveen, *Coordinating self-interested planning agents*, Autonomous Agents and Multi-Agent Systems **12** (2006), no. 2, 199–218.
9. A. J. Coles, A. I. Coles, M. Fox, and D. Long, *POPF2: a Forward-Chaining Partial Order Planner*, Procs. IPC-7, 2011.
10. Jean-Francois Cordeau and Gilbert Laporte, *The dial-a-ride problem: models and algorithms.*, Annals OR **153** (2007), no. 1, 29–46.
11. J. Cox and E. Durfee, *An efficient algorithm for multiagent plan coordination*, Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005) (Utrecht, The Netherlands), 2005, pp. 828–835.
12. Jeffrey S. Cox and Edmund H. Durfee, *Efficient and distributable methods for solving the multiagent plan coordination problem*, Multiagent and Grid Systems **5** (2009), no. 4, 373–408.
13. Mathijs M. de Weerdt and Brad J. Clement, *Introduction to planning in multiagent systems*, Multiagent and Grid Systems **5** (2009), no. 4, 345–355.
14. Daniel Delling, Thomas Pajor, and Dorothea Wagner, *Accelerating Multi-modal Route Planning by Access-Nodes*, ESA (Amos Fiat and Peter Sanders, eds.), Lecture Notes in Computer Science, vol. 5757, Springer, 2009, pp. 587–598.
15. M. desJardins, E. H. Durfee, C. L. Ortiz, and M. Wolverton, *A survey of research in distributed, continual planning*, AI Magazine **20** (1999), no. 4, 13–22.
16. M. DesJardins and M. Wolverton, *Coordinating a distributed planning system*, AI Magazine **20** (1999), no. 4, 45.
17. Y. Dimopoulos, M. A. Hashmi, and P. Moraitis, *μ-satplan: Multi-agent planning as satisfiability*, Knowledge-Based Systems **29** (2012), no. 0, 54 – 62.
18. E. Ephrati, M. E. Pollack, and J. S. Rosenschein, *A Tractable Heuristic that Maximizes Global Utility through Local Plan Combination*, In Proceedings of the First International Conference on MultiAgent Systems (ICMAS-95), 1995, pp. 94–101.
19. E. Ephrati and J. S. Rosenschein, *Multi-agent planning as a dynamic search for social consensus*, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), 1993, pp. 423–431.
20. E. Ephrati and J.S̃. Rosenschein, *Multi-agent planning as the process of merging distributed sub-plans*, In Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI 1993), 1993, pp. 115—129.
21. Eurostat, *Modal split of passenger transport*, [Online], available at tinyurl.com/eurostat-modal-split, [Accessed: Oct 26, 2012].
22. Office for National Statistics, *Special Travel Statistics (Level 1)*, [computer file], ESRC/JISC Census Programme, Census Interaction Data Service, University of Leeds and University of St. Andrews, 2001 Census.
23. D.E. Foulser, M. Li, and Q. Yang, *Theory and algorithms for plan merging*, Artificial Intelligence **57** (1992), no. 2-3, 143–181.
24. M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004.
25. M. Horn, *Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems*, Transportation Research Part A: Policy and Practice **36** (2002), no. 2, 167–188.
26. Jan Hrnčíř, *Improving a Collaborative Travel Planning Application*, Master's thesis, The University of Edinburgh, August 2011.
27. Jan Hrnčíř and Michael Rovatsos, *Applying Strategic Multiagent Planning to Real-World Travel Sharing Problems*, 7th Workshop on Agents in Traffic and Transportation, AAMAS, June 2012.
28. Chih-Wei Hsu and Benjamin W. Wah, *The SGPlan Planning System in IPC-6*, Procs. IPC-6, 2008.
29. A. Jonsson and M. Rovatsos, *Scaling Up Multiagent Planning: A Best-Response Approach*, Procs. ICAPS 2011, AAAI Press, June 2011, pp. 114–121.

30. R. Van Der Krogt, M. De Weerdt, and Y. Zhang, *Of Mechanism Design and Multiagent Planning*, Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008), IOS Press, 2008, pp. 423–427.
31. R. Larbi, S. Konieczny, and P. Marquis, *Extending Classical Planning to the Multi-agent Case: A Game-Theoretic Approach*, Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Lecture Notes in Artificial Intelligence, vol. 4724, 2007, pp. 731–742.
32. Dov Monderer and Lloyd S. Shapley, *Potential Games*, Games and Economic Behavior **14** (1996), no. 1, 124–143.
33. R. Nissim, R. Brafman, and C. Domshlak, *A general, fully distributed multi-agent planning algorithm*, Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 9, 2010, pp. 1323–1330.
34. Alejandro Torre no, Eva Onaindia, and Oscar Sapena, *An approach to multi-agent planning with incomplete information*, Proceedings of the European Conference on Artificial Intelligence (ECAI'12), 2012.
35. Thomas Pajor, *Multi-Modal Route Planning*, Master's thesis, 2009.
36. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis, *Efficient models for timetable information in public transportation systems*, Journal of Experimental Algorithmics (JEA) **12** (2008).
37. Silvia Richter and Matthias Westphal, *The LAMA planner. Using landmark counting in heuristic search*, Procs. IPC-6, 2008.
38. Frank Schulz, *Timetable information and shortest paths*, Ph.D. thesis, 2005.
39. C. Stuart, *An implementation of a multi-agent plan synchronizer*, Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85) (Los Angeles, CA), 1985, pp. 1031–1033.
40. H. Tonino, A. Bos, M. de Weerdt, and C. Witteveen, *Plan coordination by revision in collective agent based systems*, Artificial Intelligence **142** (2002), no. 2, 121–145.
41. I. Tsamardinos, M.E. Pollack, and J.F. Horty, *Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches*, Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000) (Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, eds.), 2000, pp. 264–272.
42. Roman van der Krogt and Mathijs de Weerdt, *Self-interested planning agents using plan repair*, Proceedings of the ICAPS 2005 Workshop on Multiagent Planning and Scheduling, 2005, pp. 36–44.
43. Y. Wu, L. Guan, and S. Winter, *Peer-to-peer shared ride systems*, GeoSensor Networks (2008), 252–270.
44. Q. Yang, D.S. Nau, and J. Hendler, *Merging separately generated plans with restricted interactions*, Computational Intelligence **8** (1992), no. 4, 648–676.

Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University, 121 35 Prague, Czech Republic
E-mail address: hrncir@agents.fel.cvut.cz

School of Informatics, The University of Edinburgh, Edinburgh EH8 9AB, United Kingdom
E-mail address: mrovatso@inf.ed.ac.uk

Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University, 121 35 Prague, Czech Republic
E-mail address: jakob@agents.fel.cvut.cz